Multi-Aspect Target Discrimination Using Hidden Markov Models and Neural Networks

Marc Robinson, Mahmood R. Azimi-Sadjadi, *Senior Member, IEEE*, and Jaime Salazar

***Abstract*—This paper presents a new multi-aspect pattern classification method using hidden Markov models (HMMs). Models are defined for each class, with the probability found by each model determining class membership. Each HMM model is enhanced by the use of a multilayer perception (MLP) network to generate emission probabilities. This hybrid system uses the MLP to find the probability of a state for an unknown pattern and the HMM to model the process underlying the state transitions. A new batch gradient descent-based method is introduced for optimal estimation of the transition and emission probabilities. A prediction method in conjunction with HMM model is also presented that attempts to improve the computation of transition probabilities by using the previous states to predict the next state. This method exploits the correlation information between consecutive aspects. These algorithms are then implemented and benchmarked on a multi-aspect underwater target classification problem using a realistic sonar data set collected in different bottom conditions.**

***Index Terms*—Hidden Markov models (HMMs), multi-aspect pattern classification, neural networks, prediction, underwater target classification.**

I. Introduction

In MANY target (or pattern) classification problems the availability of multiple looks at an object can substantially improve robustness and reliability in decision making. The use of several aspects is motivated by the difficulty in distinguishing between different classes from a single view at an object. It occurs frequently that returns from two different objects at certain orientations are so similar that they may easily be confused. Consequently, a more reliable decision about the presence and type of an object can be made based upon observations of the received signals or patterns at multiple aspect angles. This allows for more information to accumulate about the size, shape, composition and orientation of the objects, which in turn yields more accurate discrimination. Moreover, when the feature space undergoes changes, owing to different operating and environmental conditions, multi-aspect classification is almost a necessity in order to maintain the performance.

In general, there are three different ways of combining decisions at multiple aspects. Decision-level fusion can be used to combine, linearly or nonlinearly, the decisions, $\underline{d}_t, \underline{d}_{t-1}, \ldots, \underline{d}_{t-p}$, at several aspects where $\underline{d}_j$ represents the decision vector (output vector of the classifier) at aspect j. The sequential decision feedback system in [1] performs a combination of feature-level and decision-level fusion by generating the conditional probability $P(C_k|\underline{x}_t, \underline{d}_{t-1}, \ldots, \underline{d}_{t-p})$. In this approach, intermediate decisions are made at p previous aspects or views and once feature vector $\underline{x}_t$ at aspect t is received the final decision about the class membership of the object will be made. Feature-level fusion, on the other hand, entails computing the conditional probability $P(C_k|\underline{x}_t, \underline{x}_{t-1}, \ldots, \underline{x}_{t-p})$, based upon the previous as well as the present patterns.

A number of approaches exist to perform feature-level fusion for pattern classification. Active pattern recognition [2] and [3] utilizes multiple sensor observations in order to provide a high-confidence decision for an object by accumulating intermediate evidence over the observation period. In [2], three different framework for information fusion in active recognition systems using probabilistic, possibilistic, and Dempster-Schafer evidential theories were presented. The system dynamically repositions the camera to capture additional views in order to provide improved classification performance. It was observed that the probabilistic approach yields much better performance so long as the object-pose error rate is low. This approach implements a feature-level fusion by computing the class conditional probability using a simple method [2]. Additionally, pose or aspect estimation can be accomplished by computing the probability $P(\underline{y}_l|C_k, \underline{x}_t, \underline{x}_{t-1}, \ldots, \underline{x}_{t-N})$ where $\underline{y}_l$ represents the pose parameter vector or attributes of the object at aspect or pose l. Many of the possible fusion schemes are summarized and explained in [4] and [5]. In [6], a prediction-based algorithm that capitalizes on the correlation among the feature vectors at several consecutive aspects is introduced. The feature vectors at the previous aspects are used to predict the current feature vector using an autoregressive (AR) process. This prediction is carried out for each class and an error distribution is found based upon the training data. The likelihood of an error for a given predictor then gives a decision about the class membership of the unknown pattern when compared with the errors of the other predictors for other classes.

For many years, automatic speech recognition (ASR) community has used hidden Markov models (HMMs) for classification of a sequence of spoken phonemes [7]. The sequential nature of the HMM is very amenable to modeling the transitions that occur in speech. A single HMM is typically used to find the most likely path through the states (phonemes) and give better sequential decision than a classifier used at each phoneme. To improve the phoneme classification within the HMM model, a combination of HMM and multilayer perceptron (MLP)

Manuscript received January 29, 2003; revised December 13, 2003. This work was supported by the Office of Naval Research Biosonar Program under Contract N00014-01-1-0307.

The authors are with the Department of Electrical and Computer Engineering, Colorado State University, Fort Collins, CO 80523 USA (e-mail: azimi@engr.colostate.edu).

Digital Object Identifier 10.1109/TNN.2004.841805

network has successfully been used in ASR [11]–[13]. This combination improves the computation of emission probabilities within the HMM model by using a discriminant model. However, outside the ASR community, HMM models have not been widely applied to classification problems. In [14], HMM is used for an underwater target classification problem, where the states are defined to be a range of angles for which the features are highly correlated. The features are found using a wave-based matching pursuit algorithm [15]. The training is done with the Baum–Welch algorithm [7]. In this reference, the standard HMM formulation was used as opposed to the HMM/MLP connectionist model.

Much attention has been given to classification of sector-scan sonar imagery [8]–[10]. With these images, possible objects are detected, segmented and then classified. Often the low-quality sector-scan sonar images are filtered over a set of images, using knowledge of the movement of the sonar, to remove excess noise [8]. In another approach [9], spatial and temporal features are extracted from a set of images (interframe features) prior to classification. In [10], tracking of objects in water is done by looking at the mean optical flow of a segmented object over a set of images. Although these approaches use multiple images neither one performs a multi-aspect fusion. Additionally, unlike the acoustic backscattered data processing in this paper the operations in these references are performed on images.

In this paper, a new approach to feature-level fusion is presented using various HMM-based systems. In this framework, an accumulated decision for a sequence of patterns at several consecutive aspects is made, in which all aspects are weighted equally. This can be advantageous in situations where feature vectors collected in different environmental conditions give incorrect or inconclusive evidence about the class membership of an object. The HMM is combined with an MLP to form a more discriminant and robust classifier, and improve the multi-aspect classification performance. A new gradient descent-based method for computing the parameters of the HMM model is proposed. This gradient descent-based method uses the outputs of the MLP on the training set to define an error surface, a local minimum of which is then reached by updating the transition probabilities according to the gradient descent updating rule. Additionally, a vector prediction-based HMM structure is also introduced. This type of classifier not only uses the discriminant MLP, but also capitalizes on the underlying stochastic process of the state path within the HMM model. This method uses several predictors to predict the current state given the previous states in the observation sequence. As a result, more relevant information is used to find the higher order transition probabilities. These HMM-based algorithms are implemented and benchmarked on an underwater target classification problem using a realistic wideband sonar data set. The data was collected in different environmental conditions and, thus, is suitable for examining robustness of the proposed classifiers.

The organization of the paper is as follows. Section II gives a brief review of HMM systems and training approaches. The new gradient-based parameter training is derived in Section III. In Section IV, various connectionist approaches and their possible advantages are explained. The data set used for this paper and the preprocessing and feature extraction schemes are discussed in Section V along with the results of the various schemes. A conclusion is made in Section VI.

II. Hidden Markov Models—A Brief Review

An HMM system is typically characterized by the following quantities [7].

1) A set of states S_i, $i \in [1, N]$ that are unobservable though there is often a physical meaning attached to them.
2) A set of M observations. In a discrete HMM **[7]**, M is the number of codebook vectors, or the number of all possible observations. This implies that any observation, $\underline{v}_t$, is quantized into the set $\{\underline{x}_1, \underline{x}_2, \ldots, \underline{x}_M\}$ where $\underline{x}_m$ is the mth codebook vector.
3) A set of state "transition" probabilities represented by matrix $A = [a_{ij}]$ where $a_{ij} = P(q_t = S_j | q_{t-1} = S_i, \lambda)$ with q_t being the state visited at time t, S_i is state i and λ is the model defined by the object class and the corresponding training data.
4) A set of observation probabilities represented by matrix $B = [b_i(\underline{x}_k)]$ where $b_i(\underline{x}_k) = P(\underline{x}_k | q_t = S_i, \lambda)$ is the "emission" probability of the kth quantized observation, $\underline{x}_k$, at time t from state S_i. If the emission processes are assumed to be stationary for an HMM, the probability $b_i(\underline{x}_k)$ simply reduces to $P(\underline{x}_k | S_i, \lambda)$.
5) An initial state distribution or the probability of starting in a given state, i.e., $\pi_j = P(q_1 = S_j | \lambda)$.

Given the number of states N, and the number of observations M, the parameters A, B and π represent the model λ. There are three main issues [7] in order to maximize the performance of the HMM and identify the model in practical applications. These are briefly mentioned in the following. An in depth discussion on these topics can be found in [7].

A. Computing Model Probability

Consider a sequence of T states $Q = \{q_1, q_2, \ldots, q_T\}$ and a sequence of T observations $V = \{\underline{v}_1, \underline{v}_2, \ldots, \underline{v}_T\}$. The conditional probability of the observation sequence given the state sequence and model is $P(V|Q, \lambda) = \prod_{t=1}^{T} P(\underline{v}_t | q_t = S_i, \lambda) = \prod_{t=1}^{T} b_i(\underline{v}_t)$, where $\underline{v}_t$ is the observation at time t. In the discrete case, this observation is equivalent to $\underline{x}_k$. It is assumed that the observations are statistically independent. On the other hand, the probability of the state sequence Q given the model λ is $P(Q|\lambda) = \pi_{q_1} a_{q_1 q_2} a_{q_2 q_3} \ldots a_{q_{T-1} q_T}$. Now, using $P(V, Q|\lambda) = P(V|Q, \lambda) P(Q|\lambda)$ and summing over all possible state sequences yields

$$P(V|\lambda) = \sum_{all\ Q} \pi_{q_1} b_{q_1}(\underline{v}_1) a_{q_1 q_2} b_{q_2}(\underline{v}_2) a_{q_2 q_3} \ldots a_{q_{T-1} q_T} b_{q_T}(\underline{v}_T). \quad (1)$$

In the cases of large state sequences computing $P(V|\lambda)$, using direct implementation of (1) becomes too laborious. Thus, more efficient methods are needed [7]. The forward-backward algorithm moves through the state trellis as the number of observations increases and computes the probability of ending in a particular state at each time. This algorithm [7] is briefly reviewed here since it is used in the development of Section III.

Define the forward variable $\alpha_t(i) = P(\underline{v}_1, \underline{v}_2 \ldots \underline{v}_t, q_t = S_i|\lambda)$ that is the probability of a partial sequence of observations until time t and the state at time t given the model λ. Then, $\alpha_t(i)$ can be found iteratively using the following steps [7].

1) Initialization: $\alpha_1(i) = \pi_i b_i(\underline{v}_1)$, $\forall\ i \in [1, N]$ where $\alpha_1(i) = P(\underline{v}_1, q_1 = S_i|\lambda)$.
2) Induction: $\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(\underline{v}_{t+1})$ for $0 < t < T-1$, which is arrived at by forming $a_{ij}\alpha_t(i) = P(\underline{v}_1, \underline{v}_2, \ldots \underline{v}_t, q_t = S_i, q_{t+1} = S_j|\lambda)$, summing over all states i, and then multiplying this result by $b_j(\underline{v}_{t+1}) = P(\underline{v}_{t+1}|q_{t+1} = S_j, \lambda)$ to give $P(\underline{v}_1, \underline{v}_2, \ldots, \underline{v}_{t+1}, q_{t+1} = S_j|\lambda)$.
3) Finally, the overall probability of the observation sequence is found by $P(V|\lambda) = \sum_{i=1}^{N} P(\underline{v}_1, \underline{v}_2, \ldots, \underline{v}_T, q_T = S_i|\lambda) = \sum_{i=1}^{N} \alpha_T(i)$.

Although the backward algorithm is not needed to find $P(V|Q, \lambda)$, it will be used in the development of the optimal state sequence and the training algorithms. The backward algorithm starts at the terminal state instead of the initial state and backtracks through the state trellis with the state at time t known. Define the backward variable $\beta_t(i) = P(\underline{v}_{t+1}, \underline{v}_{t+2}, \ldots, \underline{v}_T|q_t = S_i, \lambda)$, then the steps in the backward algorithm can be summarized as [7].

1) Initialization: Set $\beta_T(i) = 1$ for all $i \in [1, N]$.
2) Induction: $\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(\underline{v}_{t+1}) \beta_{t+1}(j)$ for $t \in [1, T]$.

B. Optimal State Sequence

One approach to finding the optimal state sequence is to define a variable $\gamma_t(i) = P(q_t = S_i|V, \lambda)$, which is the probability of being in state i at time t, given the model λ and observation sequence V. Using the forward-backward variables $\alpha_t(i)$ and $\beta_t(i)$ we can write [7]$\gamma_t(i)$ as

$$\begin{aligned}\gamma_t(i) &= \frac{P(V, q_t = S_i|\lambda)}{P(V|\lambda)} \\ &= \frac{P(\underline{v}_1, \underline{v}_2 \ldots \underline{v}_t, q_t = S_i|\lambda) P(\underline{v}_{t+1}, \underline{v}_{t+2}, \ldots, \underline{v}_T|q_t = S_i, \lambda)}{P(V|\lambda)} \\ &= \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{N} \alpha_t(i)\beta_t(i)}. \end{aligned} \tag{2}$$

Then, the most likely state at time t is determined by $q_t = \arg\max_{1 \le i \le N}[\gamma_t(i)]$, $t \in [1, T]$. Although this step gives the best possible state at time t, optimizing individual states may reduce the overall probability for that path as the most likely state at each time is found regardless of the likelihood of the remaining sequence. Thus, finding the most probable sequence is often preferred to finding the best individual state and Viterbi algorithm [7], [16] is commonly used to find that most likely sequence. Instead of summing over all possible paths, Viterbi algorithm only considers the most likely path to find the best sequence. At each time t only the most likely path to each state is stored. The most likely path for each state at the next time $t+1$ is then found from the best path at time t.

C. Maximization of $P(V|\lambda)$

Unfortunately, there is no optimal way to maximize $P(V|\lambda)$, as there is no analytical solution for the problem of maximizing the probability of an observation sequence. However, there are several methods [7] that can locally maximize $P(V|\lambda)$ using iterative approaches. The most commonly used approaches are the Baum–Welch and gradient descent [17] methods. The Baum–Welch training procedure uses the forward-backward variables and is treated in length in [7]. In Section III, we present a new gradient descent-based approach for this step.

III. Gradient Descent for HMM Parameterization

One approach to find optimum parameters a_{ij} and $b_j(\underline{v}_i)$ is by defining a cost function and using the gradient descent method [17] to yield an iterative algorithm for parameter updating. The cost function used here is

$$J = -\log(P(V|\lambda)) \tag{3}$$

where $P(V|\lambda) = \sum_{i=1}^{N} \alpha_T(i)$. If θ is one of the parameters in the set λ, J is minimized by the recursive application of

$$\theta_n = \theta_{n-1} - \eta \left[\frac{\partial J}{\partial \theta}\right]_{\theta=\theta_{n-1}} \tag{4}$$

where η is the step size and $\partial J/\partial\theta = -(1/P(V|\lambda))(\partial P(V|\lambda)/\partial\theta)$.

Let $\underline{\alpha}_t = [\alpha_t(1)\ \alpha_t(2)\ \ldots\ \alpha_t(N)]$, with initial value $\underline{\alpha}_1 = [\pi_1 b_1(\underline{v}_1)\ \pi_2 b_2(\underline{v}_1)\ \ldots\ \pi_N b_N(\underline{v}_1)]$, then we can write $P(V|\lambda) = \sum_{i=1}^{N} \alpha_T(i) = \underline{\alpha}_T\, \underline{1}_{N\times 1}$ where $\underline{1}_{N\times 1}$ is a column vector of N ones. Using the induction step in the forward pass we can define a matrix $C(t)$ for which $\underline{\alpha}_t = \underline{\alpha}_{t-1} C(t)$ where the elements of $C(t)$ are $c_{ij}(t) = a_{ij} b_j(\underline{v}_t)$. Iterative application of the equation for $\underline{\alpha}_t$, after T iterations gives $\underline{\alpha}_T = \underline{\alpha}_1 \prod_{k=2}^{T} C(k)$. Similarly, if we define $\underline{\beta}_t = [\beta_t(1)\ \beta_t(2)\ \ldots\ \beta_t(N)]^t$ with terminal value $\underline{\beta}_T = \underline{1}_{N\times 1}$, we get an iterative equation $\underline{\beta}_t = C(t+1)\underline{\beta}_{t+1}$ leading to $\underline{\beta}_t = \prod_{k=t+1}^{T} C(k)\underline{\beta}_T$. To update the parameters a_{ij}, we need $(\partial P(V|\lambda)/\partial a_{ij})$. The previous expression for $P(V|\lambda)$ can be used to yield

$$\begin{aligned}\frac{\partial P(V|\lambda)}{\partial a_{ij}} = \sum_{k=2}^{T} &\underline{\alpha}_1 C(2) \ldots C(k-1) \\ &\times \frac{\partial C(k)}{\partial a_{ij}} C(k+1) \ldots C(T) \underline{1}_{N\times 1}. \end{aligned} \tag{5}$$

Using the definitions of the forward and backward variables, (5) can be rewritten

$$\frac{\partial P(V|\lambda)}{\partial a_{ij}} = \sum_{k=2}^{T} \underline{\alpha}_{k-1} W_{ij}(k) \underline{\beta}_k b_j(\underline{v}_k) \tag{6}$$

where $W_{ij}(k)$ is a matrix that has only one nonzero element at its (i, j) entry, i.e., $w_{ij} = 1$. Thus, we can write $W_{ij}(k) = \underline{e}_i^t\, \underline{e}_j$ where $\underline{e}_k = [0 \ \ldots \ \underset{k}{1} \ 0 \ \ldots]$. Now, using this property in (6) the updating equation (4) for a_{ij}'s becomes

$$\overline{a}_{ij} = a_{ij} - \eta \left[\frac{1}{\underline{\alpha}_T\, \underline{1}_{N\times 1}} \sum_{k=2}^{T} \alpha_{k-1}(i) \beta_k(j) b_j(\underline{v}_k)\right]. \tag{7}$$

The updating equation for $b_j(\underline{v}_i)$ can similarly be obtained by expressing

$$\frac{\partial P(V|\lambda)}{\partial b_j(\underline{v}_i)} = \sum_{k=2}^{T} \underline{\alpha}_1 C(2)\ldots C(k-1)\frac{\partial C(k)}{\partial b_j(\underline{v}_i)} \times C(k+1)\ldots C(T)\underline{1}_{N\times 1} + \frac{\partial \underline{\alpha}_1}{\partial b_j(\underline{v}_i)} C(2)C(3)\ldots C(T)\underline{1}_{N\times 1}. \quad (8)$$

The first term $\partial C(k)/\partial b_j(\underline{v}_i) = \mathbf{0}$ for all $k \neq i$. For $k = i$, this partial derivative yields a matrix with only one nonzero column, i.e., the jth column that is $[a_{1j}\ a_{2j}\ \ldots\ a_{Nj}]^t$. Additionally, the second term in the right side of (8) is always zero $\forall\, i \neq 1$. Thus, for these values of i, using the induction step in Section II for the forward variable $\alpha_i(j)$, (8) can be simplified to

$$\begin{aligned}\frac{\partial P(V|\lambda)}{\partial b_j(\underline{v}_i)} &= \underline{\alpha}_{i-1}\,[a_{1j}\ a_{2,j}\ \ldots\ a_{Nj}]^t \beta_i(j) \\ &= \left[\sum_{k=1}^{N} \alpha_{i-1}(k)a_{kj}\right]\beta_i(j) \\ &= \frac{\alpha_i(j)\beta_i(j)}{b_j(\underline{v}_i)} \quad \forall\, i \in [2,T]. \end{aligned} \quad (9)$$

For $i = 1$, we only need to consider the second term in (8), i.e., $(\partial\underline{\alpha}_1/\partial b_j(\underline{v}_1))C(2)C(3)\ldots C(T)\underline{1}_{N\times 1}$ where $\partial\underline{\alpha}_1/\partial b_j(\underline{v}_1) = \pi_j\ \underline{e}_j$ and $C(2)C(3)\ldots C(T)\underline{1}_{N\times 1} = \underline{\beta}_1$ since $\underline{\beta}_T = \underline{1}_{N\times 1}$. Hence, the second term is $\pi_j\beta_1(j)$ or equivalently $(\alpha_1(j)\beta_1(j)/b_j(\underline{v}_1))$. Thus, for all $i \in [1,T]$ the updating equation for $b_j(\underline{v}_i)$ becomes

$$\overline{b}_j(\underline{v}_i) = b_j(\underline{v}_i) - \eta\left[\frac{1}{\underline{\alpha}_T\underline{1}_{N\times 1}}\frac{\alpha_i(j)\beta_i(j)}{b_j(\underline{v}_i)}\right]. \quad (10)$$

Alternatively, the batch mode training algorithm finds the gradient over the entire training set before updating. In this case, a different cost function J is defined, $J = -\log\left[\prod_l P(V^l|\lambda)\right] = -\sum_l \left[\log P(V^l|\lambda)\right]$ where V^l is the lth observation sequence in the training set and $\underline{v}_k^l$ is the kth observation in that sequence. Similarly, $\underline{\alpha}_T^l$ and $\underline{\beta}_k^l$ are the forward and backward variables found in the lth sequence. Following a similar approach as with the online method, the updating rule for a_{ij} becomes

$$\overline{a}_{ij} = a_{ij} - \eta\left[\sum_l \frac{1}{\alpha_T^l\ \underline{1}_{N\times 1}}\sum_{k=2}^{T}\alpha_{k-1}^l(i)\beta_k^l(j)b_j\left(v_k^l\right)\right] \quad (11)$$

which sums over all sequences. Similarly, for the parameters $b_j(\underline{v}_i)$, the batch mode updating equation is

$$\overline{b}_j(\underline{v}_i) = b_j(\underline{v}_i) - \eta\left[\sum_{l\ and\ \underline{v}_i}\frac{1}{\alpha_T^l\ \underline{1}_{N\times 1}}\frac{\alpha_i^l(j)\beta_i^l(j)}{b_j\left(v_i^l\right)}\right]. \quad (12)$$

In the discrete case this equation is greatly simplified by the fact that $\underline{v}_i$ takes values in the set $\{\underline{x}_1, \underline{x}_2, \ldots, \underline{x}_M\}$. The equation is then a simple summation over all $\underline{x}_k$ that occur in all observation sequences in the training set. Using these updating equations, it is possible to find a local minimum in the error surface that gives better performance over the Baum–Welch training algorithm.

IV. Connectionist Approaches

A. Discriminant Local Probability

There are several drawbacks to the discrete HMM [12]. First and foremost is the assumption of the discrete nature of the observations that requires applying some type of vector quantization on the continuous data. However, quantization of such quantities as features of sonar data results in significant loss of information. A possible solution is a continuous HMM which uses multiple multivariate Gaussian distributions to model the feature space [18]. Clearly, this requires a complete knowledge of the statistical properties of the data set in the feature space. Additionally, for a large dimensional feature space it is difficult to know how well multiple multivariate Gaussians model the data.

The second drawback is the first-order Markov assumption as the state sequence is often governed by a higher order process. This will be addressed in Section IV-C. The third drawback has to do with the fact that the model is not discriminatory where the term "discriminatory"[11] in this case is used to describe a model that uses all available information to make a decision. The HMM model is not discriminatory as the local probability

$$\begin{aligned} a_{ij}b_j(\underline{v}_{t+1}) &= P(q_{t+1} = S_j|q_t = S_i, \lambda)P(\underline{v}_{t+1}|q_{t+1} = S_j, \lambda) \\ &= P(q_{t+1} = S_j|q_t = S_i, \lambda) \\ &\quad \times P(\underline{v}_{t+1}|q_{t+1} = S_j, q_t = S_i, \lambda) \\ &= P(\underline{v}_{t+1}, q_{t+1} = S_j|q_t = S_i, \lambda) \end{aligned}$$

uses the *a priori* emission probability and not the observation at time $t+1$. A discriminant local probability, $a_{ij}b_j(\underline{v}_{t+1}) = P(q_{t+1} = S_j|\underline{v}_{t+1}, q_t = S_i, \lambda)$, that uses the knowledge of the current observation $\underline{v}_{t+1}$, should improve the performance of the overall HMM system. This can be done by computing the *a posteriori* probability, $P(q_{t+1} = S_j|\underline{v}_{t+1}, \lambda)$, and then using the Bayes rule to find the emission probability, i.e., $P(\underline{v}_{t+1}|q_{t+1} = S_j, \lambda) = (P(q_{t+1} = S_j|\underline{v}_{t+1}, \lambda)P(\underline{v}_{t+1}|\lambda)/P(q_{t+1} = S_j|\,\lambda))$. The product of the emission and transition probabilities will then generate the desired discriminant local probability.

In the ASR community, an MLP network is typically trained to find the *a posteriori* probability for a given set of known states. Here, the observations are considered to be equally likely, i.e., $P(\underline{v}_{t+1}|\lambda)$ is constant across all models and similarly for the probability $P(q_{t+1} = S_j|\lambda)$. The inputs (observations) to the MLP can be continuous valued feature vectors. Fig. 1 shows the procedure for finding the local discriminant probabilities using the HMM/MLP system. The process is repeated for each new observation using the Viterbi method to find the most likely states, and the product of the local probabilities gives the overall probability for that sequence.

There are two possible training schemes for the MLP [19] used in conjunction with the HMM. The first approach [12] is to fix the definition of the states, where in ASR each state corresponds to a spoken phoneme, in which case the model is no longer "hidden." The MLP is trained to give $b_j(\underline{v}_t)$, and then the parameters a_{ij} are subsequently computed. The MLP must be trained on a given training data set, with the number

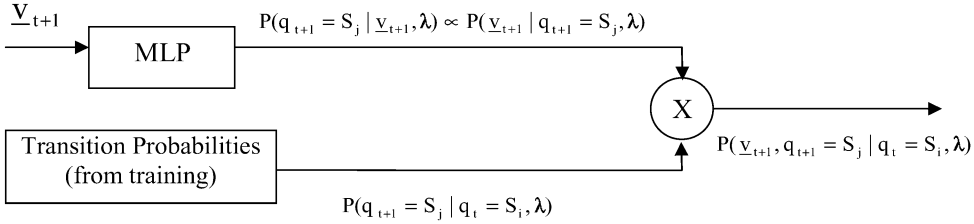

Fig. 1. Block diagram of the HMM/MLP system for local probabilities.

of states as the number of output neurons i.e each output finds $P(q_t = S_i|\underline{v}_t)$ for all $i \in [1, N]$ which is used for all models λ. Finding the maximum probability across all outputs as the state at time t, state transitions are counted and used as initial estimates of the parameters a_{ij}. The training of a_{ij} may be done in many ways, e.g., Viterbi algorithm [7] and gradient descent [17].

The second approach requires the assumption that the states are not defined and allows the training algorithm to find the states and transitions through an iterative training during which the states are hidden and may not have any physical meaning. An initial state sequence is decided on the training data and the MLP is first trained to correspond to that state sequence. Using either the Baum–Welch or the gradient descent in Section III, the parameters a_{ij} are then updated using the emission probabilities found by the trained MLP. The updated parameters, $b_j(\underline{v}_t)$ and a_{ij}, are used to find new values of the parameters $\alpha_t(i)$, $\beta_t(i)$ using the iterations found in Section II-A and $\gamma_t(i)$ using (2). These are then used to find a new set of $b_j(\underline{v}_t)$ by using either (10) or (12). The MLP is then retrained to provide the new $b_j(\underline{v}_t)$. This process is repeated until a convergence criterion is met.

B. HMM/MLP in Sonar Classification

In our underwater target classification application, the desired state sequence for a given observation sequence stays in one state since the states represent objects. Thus, a single MLP is used to estimate the state for the current observation. Although this network may not exactly mimic the discriminant HMM [11], it yields the discriminant local probabilities $P(q_{t+1} = S_j|\underline{v}_{t+1}, \lambda)$. This allows the HMM/MLP to become discriminatory as well as to overcome the discrete nature of the HMM, though it cannot circumvent the issues with first-order Markov assumptions. One approach to overcome the latter is to use two sets of MLPs to imitate an HMM, where each MLP network estimates a set of HMM parameters, $(a_{ijk}, b_j(\underline{v}_t))$. In this case, a_{ijk} is a higher order transition probability found for each class. The outputs of the two models are compared and the model with the highest $P(V|\lambda)$ decides the class membership. This approach is discussed in Section IV-C.

C. Prediction-Based HMM

As mentioned before, if the MLP is trained to generate the emission probabilities, the outputs are expected to provide very good evidence about the state at time $t + 1$. Let the MLP output vector at time $t + 1$ be $\underline{z}_{t+1} := [P(q_{t+1} = S_1|\underline{v}_{t+1}, \lambda) \ \ldots \ P(q_{t+1} = S_N|\underline{v}_{t+1}, \lambda)]^t$. Using p previous vectors $\{\underline{z}_t, \ldots, \underline{z}_{t-p+1}\}$, one can generate the minimum variance (MV) prediction of $\underline{z}_{t+1}$. This corresponds to conditional expectation, $\hat{\underline{z}}_{t+1} = E[\underline{z}_{t+1}|\underline{z}_t, \ldots, \underline{z}_{t-p+1}, \lambda]$, where λ represents the model and is defined by the number of states and the training set. The prediction error vector $\underline{e}_{t+1}$ is defined by $\underline{e}_{t+1} = \underline{z}_{t+1} - \hat{\underline{z}}_{t+1}$. The prediction can be done in many ways. However, if Markov process of order p is assumed, the linear MV prediction can be performed using a vector AR model

$$\hat{\underline{z}}_{t+1} = A_1\underline{z}_t + A_2\underline{z}_{t-1} + \cdots + A_p\underline{z}_{t-p+1} \tag{13}$$

where A_i's, $i \in [1, p]$ are parameter matrices to be estimated. This formulation assumes that the discriminant probabilities (or equivalently the emission probabilities $b_i(\underline{v}_{t+1})$) for a given model (class) follow a vector AR model.

In this problem, since states correspond to objects, we could simply use the probabilities of the state vector at previous aspects to predict the next state. In this framework, similar state (object) transition scenarios, such as those found within a model for one class, yield almost equal prediction error. Thus, the distribution of the prediction error can be used to represent the likelihood of different state transitions in several consecutive aspects. The distribution of this prediction error vector $\underline{e}_{t+1}$ is assumed to be multivariate Gaussian for each model with a zero mean vector and covariance matrix Σ_λ determined based on the training data set. Thus, we have

$$p(\underline{e}_{t+1}|\lambda) = \frac{1}{(2\pi)^{(N/2)}(\det(\Sigma_\lambda))^{(1/2)}} \times \exp\left(-\frac{1}{2}\underline{e}^t_{t+1}\Sigma_\lambda^{-1}\underline{e}_{t+1}\right) \tag{14}$$

which is then normalized by the sum of $p(\underline{e}_{t+1}|\lambda)$ across all λ's. Although, the multivariate Gaussian assumption may not give an exact fit to the error distribution, it is a reasonable assumption and further it simplifies the development.

Now, let us reconsider the HMM equation in Section II, i.e., $P(V|\lambda) = \sum_{all\ Q} P(V|Q, \lambda)P(Q|\lambda)$ and generalize some of the previous assumptions. In particular, if the assumption of the first-order Markov process is generalized, the term $P(Q|\lambda)$ can be expressed as

$$P(Q|\lambda) = P(q_T|q_1, \ldots, q_{T-1}, \lambda) \times P(q_{T-1}|q_1, \ldots, q_{T-2}, \lambda)\ldots P(q_2|q_1, \lambda)P(q_1|\lambda). \tag{15}$$

Now, if a pth-order Markov model is assumed, i.e., q_t depends only on the past p states, then (16) becomes

$$P(Q|\lambda) = P(q_T|q_{T-p}, \ldots, q_{T-1}, \lambda) \times P(q_{T-1}|q_{T-p-1}, \ldots, q_{T-2}, \lambda)\ldots P(q_2|q_1, \lambda)P(q_1|\lambda). \tag{16}$$

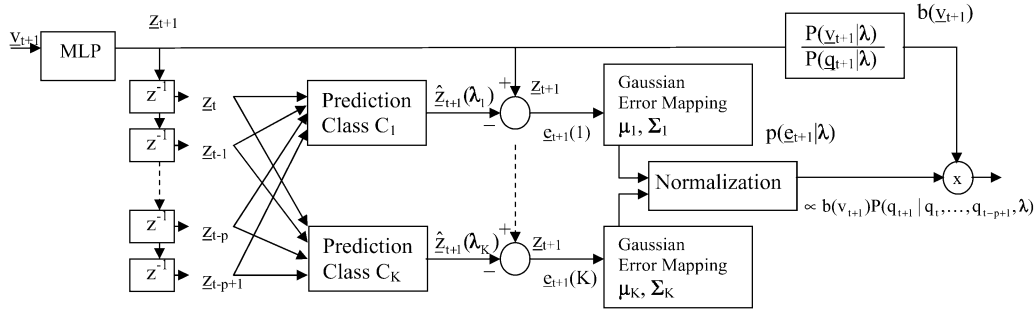

Fig. 2. Block diagram of the prediction algorithm for local probabilities.

The term $P(V|Q,\lambda)$ in the expression for $P(V|\lambda)$ does not change i.e, $P(V|Q,\lambda) = \prod_{t=1}^{T} P(\underline{v}_t|q_t = S_i,\lambda) = \prod_{t=1}^{T} b_i(\underline{v}_t)$, which can be computed as described in Section IV-A. Now using $r_p(t+1) = p(\underline{e}_{t+1}|\lambda) = p(\underline{z}_{t+1}|\underline{z}_t,\ldots,\underline{z}_{t-p+1},\lambda) \propto P(q_{t+1}|q_t,\ldots,q_{t-p+1},\lambda)$ in (16) in place of high-order transition probabilities and combining with the terms in $P(V|Q,\lambda)$ yields

$$P(V|\lambda)=\sum_{all\ Q} \pi_{q_1} b_{q_1}(\underline{v}_1) r_p(2) b_{q_2}(\underline{v}_2) r_p(3)\ldots r_p(T) b_{q_T}(\underline{v}_T). \tag{17}$$

One drawback of this system is that the number of previous aspects available for the predictor is not always fixed at p and it can change from 1 to p. To circumvent this problem, a bank of predictors (for 1 up to p previous vectors) can be trained. Using this system, predictions can be made for any number of available previous vectors between 1 to p. Nevertheless, the problem is that predictions based upon only a few previous aspects will not typically give good results.

Now, if we use $r_p(t+1)$ as the transition probability in the HMM/MLP model with the *a posteriori* emission probabilities found by the MLP we get local probabilities $b(\underline{v}_{t+1}) r_p(t+1) \propto P(q_{t+1}|\underline{v}_{t+1},\lambda) P(q_{t+1}|q_t,\ldots,q_{t-p+1},\lambda)$. This is proportional to the discriminant local probabilities discussed earlier. In this multi-aspect target classification problem, there could either be prediction models for each object or for each class. Both approaches were tried and the latter approach was found to provide better performance, especially when feature vectors from different environmental conditions are encountered.

Fig. 2 depicts the process of finding the local probabilities in which the *a posteriori* emission probability provided by the MLP are applied, via a tapped delay line, to the state predictors, which estimate the current state vector. The prediction error is then mapped to the high-order transition probabilities, which are then combined with the emission probability at the current aspect to give the local discriminant probabilities. The product of each of the local probabilities then provides the final result $P(V|\lambda)$ as in (17). Note that each class has its own model for prediction.

To find the parameter matrices, A_i's, $i \in [1,p]$, in the vector AR model in (13) the method in [20] was used. In this reference, a single-layer network with linear neurons and a recursive least squares (RLS)-based learning method was used for parameter estimation of a linear vector AR predictor. The system uses the RLS learning rule to find the MV estimate of the parameters in the linear vector predictor in (13).

Let us define the augmented input vector to the predictor $\underline{\mathbf{Z}}(t) = \left[\underline{z}_t^t, \underline{z}_t^t, \ldots, \underline{z}_{t-p+1}^t\right]^t$ for the observation sequence up to time t, and $\underline{\mathbf{W}}_n = [w_{1,n}(1), w_{2,n}(1), \ldots, w_{N,n}(1), \ldots, w_{1,n}(p), \ldots, w_{N,n}(p)]^t$ as the weight vector for the nth output node. Thus, the actual output at node n is $\hat{z}_{t+1}(n) = \underline{\mathbf{Z}}_n^t \underline{\mathbf{U}}(t)$ that corresponds to the nth element of $\underline{\hat{z}}_{t+1}$; while the desired output for this node is $z_{t+1}(n)$, i.e., the nth element of $\underline{z}_{t+1}$. The goal of the learning is to iteratively find the best $\underline{\mathbf{W}}_n$'s such that $\hat{z}_{t+1}$ approaches z_{t+1} in the MV error sense. A vector AR model is identified for each class or model λ using the RLS learning rule [21]. In our target classification problem, the training set consists of various sequences of $\{\underline{z}_t,\ldots,\underline{z}_{t-p+1}\}$ for different aspect separation intervals. At convergence $\underline{\mathbf{W}}_n \longrightarrow [a_{n,1}(1)\ldots a_{n,N}(1),\ldots a_{n,1}(p)\ldots a_{n,N}(p)]^t$ where $a_{n,i}(j)$ is the (n,i)th element of matrix A_j in the vector AR process. Once converged, the training data for each class are applied as inputs to the class predictor and the outputs are used to generate error vectors $\underline{e}_{t+1} = \underline{z}_{t+1} - \underline{\hat{z}}_{t+1}$ for all times t, which will in turn form the error distribution.

V. Test Results and Discussions

The proposed multi-aspect feature-level fusion algorithms are tested on a data set collected at Applied Research Lab, University of Texas, (ARL-UT) and Lake Travis Test Station (LTTS) facilities. Acoustic backscattered signals were acquired for three mine-like and three non-mine-like objects. The mine-like targets consisted of two metallic cylindrical objects of different sizes and a plastic truncated cone shaped object placed on the flat side. The non-mine-like objects included a steel drum, concrete pipe and a 6-ft section of a telephone pole. The data sets were collected in different environmental conditions namely free-field, smooth or rough bottom sand and gravel bottom condition. In this study, the data sets for smooth and rough bottom conditions were used.

Fig. 3 shows the data collection setup for different bottom conditions. In these cases, the objects were placed on a rotating seabed, with a diameter of 25 ft, 25–30 ft below the surface of the lake. The center of the target is positioned as near the center of the circular platter as possible. Certain straps and parts of the supporting barge are also in the water and can cause secondary returns. The setup at the LTTS allows for precise knowledge and

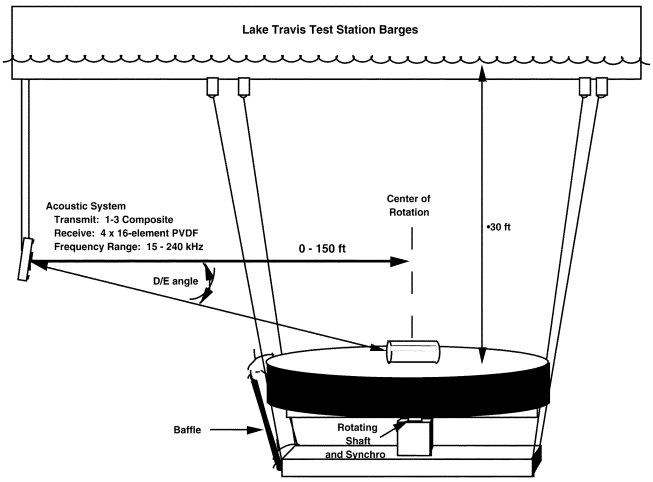

Fig. 3. Experimental bottom setup for the wideband data set.

control of aspect angles. The details of the experimental conditions are described in a document provided by ARL-UT [22]. The rough bottom condition is similar to the smooth bottom, but the sand is raked, giving it a rippled effect. The target is rotated in a horizontal plane while the acoustic panel is set at a fixed depression/elevation (D/E) angle. A single rotation is executed while backscattered acoustic waveforms at nearly uniform 1° increments are collected. The data was collected at a range of 105 ft. with a D/E angle of either 3.5°, 8.5°, or 13.5° for several mine-like and non-mine-like objects in the different environmental conditions. The data set for D/E of 13.5° was used for this study.

The data set has been collected using a receiver with an array configuration of 16 channels, with separation of 1 in. A linear frequency modulated (LFM) transmit waveform with 7-ms duration was used with a frequency bandwidth of 85 kHz in the range of 15kHz–100 kHz. In this study, the data of four channels were averaged together to yield a beamwidth that is just enough to insonify the object. Beamwidth (in degrees) is computed approximately using 50/aperture length (in wavelengths). At 100 kHz (i.e., wavelength $= 0.6$ in) the length of the four channel aperture is 6.6 wavelengths, giving a beamwidth of $50/6.6 = 7.6^\circ$. At a distance of 105 ft, the beamwidth gives us a coverage width of about 14 ft, which is wide enough to cover the entire length of the object and short enough to avoid the side edges of the seabed. The sonar returns are 16 msec long and sampled at a sampling frequency of 500 kHz resulting in 8192 samples. The transmit signal was also sampled at 500 kHz, resulting in 4096 samples.

Due to the experimental setup [22], the presence of artifacts caused by the barge, seabed, etc. is inevitable in the backscattered signals. Although these artifacts are often in front of, or lag behind the main returns, at certain aspects they overlap with the main return. To overcome this problem, an inverse matched filtering algorithm is adopted [23]. This method primarily relies on windowing in the matched filtered domain and then performing inverse filtering to recover the "clean backscattered" signal. The method exploits the fact that the artifact is more separated from the main target return in the matched filtered domain than in the time domain. After the "clean" signal is obtained via windowing in the matched filtered domain, several (28) subband features that represent certain tonal attributes of each signal are extracted using wavelet packets and linear predictive coding schemes [23]. The dynamic range of the subband features for this data set is somewhat large. These very large features can cause problems for classifiers when the feature vectors vary significantly from those presented during the training. To avoid this problem each feature vector was normalized by its norm.

In each of the following studies, training, validation and testing sets remain consistent. The training set consisted of every fourth aspect of the smooth bottom condition (90 aspects for each object). The validation set was the remaining smooth bottom aspects and the testing was formed of all the rough bottom data (360 aspects for each object). The role of the validation set in this context is a larger set in the same environmental condition as the training set used to test the generalization of the trained classifier. Thus, the validation set was not used to pick any decision threshold or the best network, although it could also be used for this purpose. The testing set, on the other hand, can help us to determine the robustness to environmental variations and, hence, the usefulness of the system in real settings. Sequences of aspects for the training set were formed with ten different aspect separations. The separation was varied from 4° to 40°. The validation and testing sets were generated with an aspect separation of 8° plus a uniform discrete r.v. on $[-4^\circ, 3^\circ]$, so that two consecutive observations are not the same aspect. For each observation sequence there are ten different realizations of aspect separation for both the validation and the testing sets. Thus, the validation set has 10800 sequences and the testing set has 14400 sequences. Note that the random aspect separation was implemented to best imitate actual mine-hunting scenarios where due to nonuniform motion of the vehicle carrying the sonar, aspect separation variations can occur. In realistic situations a towed or autonomous underwater vehicle makes a single pass by an object in which multiple looks at the object are received. These observations normally cover between 20%–40% of the entire object. Thus, with ten observations, a maximum of 80° plus the r.v on $[-4^\circ, 3^\circ]$ of aspects will be seen that is less than 25% of the object. Test results are presented for discrete HMM, HMM/MLP with Viterbi training, the new online gradient descent, batch gradient descent and the prediction-based HMM.

A. Discrete HMM Using Baum–Welch Training

To train a discrete HMM some sort of vector quantization needs to be performed. In this study, the k-means algorithm [24] was used to form k clusters. The clustering of the k-means algorithm is done without supervision, only the number of clusters is known. This process starts by randomly choosing k vectors from the set of samples in the training set $X = \{\underline{x}_1, \underline{x}_2, \ldots, \underline{x}_N\}$, and using these k samples as centroids of k clusters. Each vector is then associated with one of the k clusters represented by their means. This association is done by a distance measure. The mean of each cluster is then recalculated based upon the vectors associated with that cluster. The process is repeated until there are no further changes in the

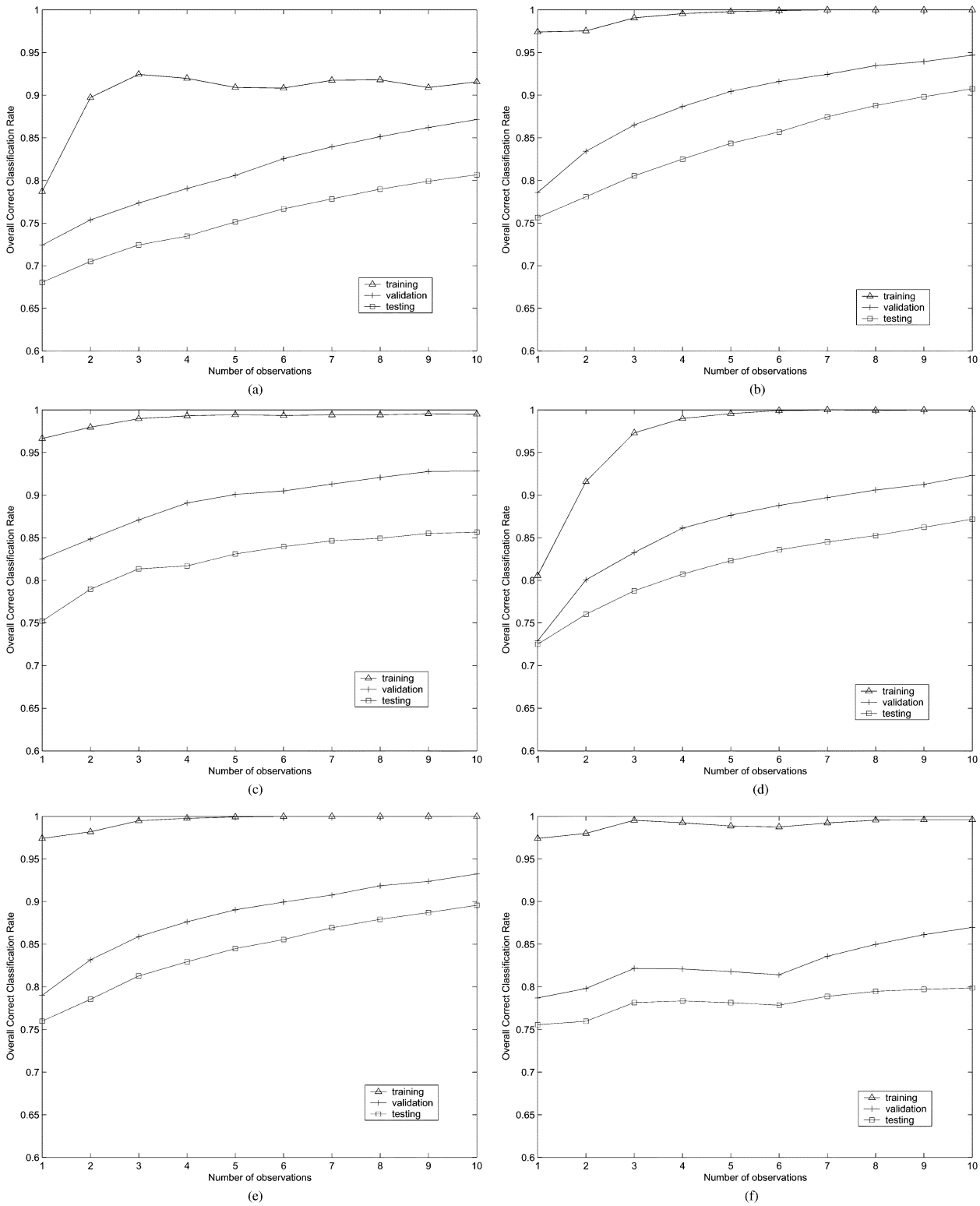

Fig. 4. Overall correct classification rates versus observation length for different multi-aspect fusion schemes. (a) Discrete HMM. (b) MLP with HMM using Viterbi alignment. (c) Sequential decision feedback. (d) MLP with HMM using gradient descent. (e) MLP with HMM using batch gradient descent. (f) Prediction-based HMM.

clusters. For this study 135 codebook vectors were specified. Two HMMs were trained, one for the mine-like objects and the other for non-mine-like objects. Each discrete HMM was trained using the Baum–Welch algorithm [7]. The number of states chosen was six, corresponding to the number of objects that are in the data set. The sequence probability computed from each HMM are compared and the class label associated with the greatest probability is selected. The results of the discrete HMM for overall correct classification rate versus observation length (number of aspects) are shown in Fig. 4(a). As can seen,

the general trend shows that as the observation size increases the performance improves. The trend is slightly slower for the testing set than the validation. This is most likely caused by the different environmental (rough bottom) condition. Clearly, the results of the discrete HMM are poor on both the validation and testing sets even after ten observations.

B. HMM/MLP With Viterbi Training

When a fixed MLP is used to estimate the emission probabilities, no vector quantization is needed. A three-layer MLP was trained to perform single aspect state probability calculations, with the states being the six different objects. The best trained MLP network has a structure 28-60-28-6 and generates the conditional probability of each state given the feature vector, i.e., $P(q_{t+1} = S_i|\underline{v}_{t+1}, \lambda)$. The maximum probability among the six outputs is used to determine the class of the unknown object in the two-class problem. The system correctly classifies the training, validation and testing sets at 97.2%, 78.6%, and 75.6%, respectively, for single aspect. These results using the single aspect classification indeed attest to a mixed feature space as the classes are not separable by the decision surface, even for the training data. Based upon the emission probabilities $b_j(\underline{v}_t)$ generated by the MLP, the transition probabilities were updated using a forced Viterbi alignment as described in [7]. The convergence of the parameters are shown in Fig. 5. The parameters for the non-mine-like model converge only after two epochs, while the parameters for the mine-like objects require five epochs to converge.

The performance plots of the overall HMM/MLP system are shown in Fig. 4(b). It is easily seen that the results based on the first aspect are much better than those of the discrete HMM case. This is due to the MLPs ability to classify single aspects of data well. After ten aspects, the HMM/MLP provides 95% and 91% overall correct classification rates on the validation and testing sets, respectively. These results show drastic improvements over those of the discrete HMM, and also a small improvement (2% and 4% on the validation and testing sets, respectively) over the sequential decision feedback [1] in Fig. 4(c), which implements a combination of the decision-level and feature-level fusion. It is interesting to note that the trend is still slightly upward even after ten aspects, implying that even better performance could be achieved with more than ten aspects.

C. HMM/MLP With Gradient Descent Training

1) Online Learning: Here the same MLP is used for computing the emission probabilities. However, instead of using the Viterbi training, the transition probabilities are found using the gradient descent scheme proposed in Section III. The initial values of the parameters $b_j(\underline{v}_{t+1})$ are found for all t by using the Viterbi paths across each training sequence and counting the transitions within the paths. This method finds a local minimum in the error surface over all sets of observations. In this case, a learning rate $\eta = 0.001$ was used with the maximum number of epochs of 100. The convergence of the parameters is shown in Fig. 5. After two epochs, the change in parameters is less than 10^{-5} for both models (or classes). The performance plots of this system are shown in Fig. 4(d). As evident from these plots, the online gradient descent gives results

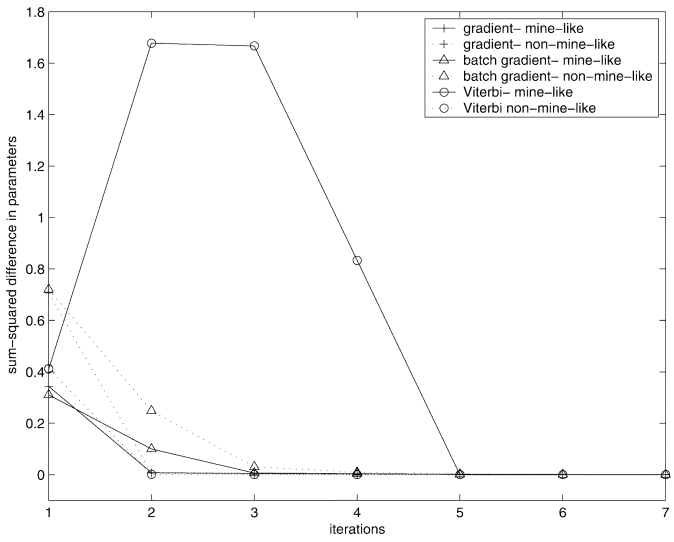

Fig. 5. Convergence of HMM parameters for the different HMM/MLP schemes.

that are very close to those of the Viterbi algorithm. The Viterbi training actually shows a slight improvement over the gradient descent method (less than 3%) after ten observations on both the validation and testing sets. Both of these HMM/MLP-based approaches are far superior over the standard discrete HMM approach with Baum–Welch training.

2) Batch Learning: The batch gradient descent training in Section III was then implemented based upon the outputs of the MLP ($b_j(\underline{v}_{t+1})$) using (11) for generating the transition probabilities a_{ij}. Again, the initial values of $b_j(\underline{v}_{t+1})$ are found by the Viterbi paths in the training sequences. The parameters were the same as in the online case, namely $\eta = .001$ with 100 as the maximum number of epochs. The convergence of the parameters is shown in Fig. 5. After five epochs, the change in parameters is less than 10^{-4} for both models. This approach is somewhat slower to converge than the other HMM/MLP training algorithms. The performance plots of this system are shown in Fig. 4(e). As can be seen, the performance of the batch gradient descent is very similar to the Viterbi training. The batch method performs about 2% better than the online approach at all sizes of observation sequences. This is likely caused by the fact that for the same learning rate the online version will be more susceptible to misadjustment. The batch gradient descent method has a better performance on the testing set (by about 4%) than the sequential decision feedback system, while they are about the same on the validation set after ten aspects.

For comparison purposes, the performance plots of the Viterbi training and batch gradient training are presented together in Fig. 6. As can be observed, the batch gradient descent method has a slightly better performance on the testing data set than the Viterbi training for less than six aspects and a slightly worst performance for more than six aspects. Moreover, the gradient descent converges to a local minimum in an error surface, while the Viterbi algorithm has no such guarantee and it is possible that the training will not yield good estimates for a_{ij}'s. To see this, it is interesting to compare the transition probabilities computed by each method. Table I presents the elements of the transition probability matrices found by the

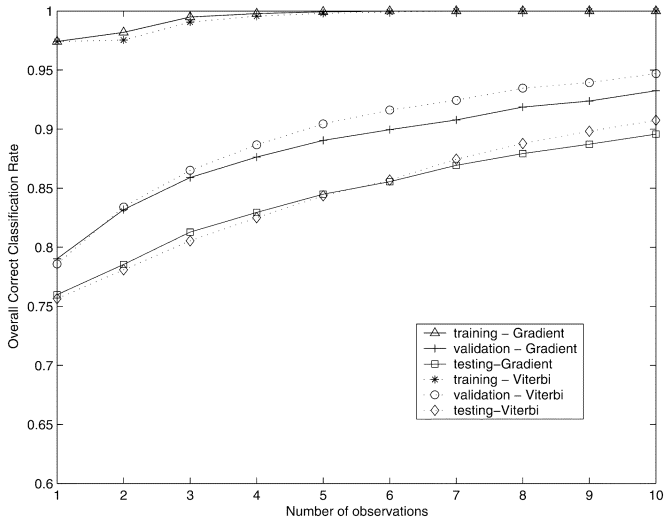

Fig. 6. Comparison of Viterbi training and batch gradient descent—overall correct classification versus observation length.

TABLE I
TRANSITION MATRICES AFTER VITERBI AND BATCH GRADIENT DESCENT TRAINING

Viterbi Mine-Like Transition Probabilities						
State	1	2	3	4	5	6
1	.98	.01	ϵ	ϵ	.01	ϵ
2	.01	.99	ϵ	ϵ	ϵ	ϵ
3	ϵ	ϵ	1	ϵ	ϵ	ϵ
4	.17	.17	.17	.17	.17	.17
5	1	ϵ	ϵ	ϵ	ϵ	ϵ
6	.17	.17	.17	.17	.17	.17

Viterbi Non-Mine-Like Transition Probabilities						
State	1	2	3	4	5	6
1	ϵ	.03	ϵ	ϵ	.97	ϵ
2	ϵ	.07	ϵ	.01	.93	ϵ
3	.17	.17	.17	.17	.17	.17
4	.01	.01	ϵ	.98	.01	ϵ
5	.03	.03	ϵ	.01	.93	.01
6	ϵ	ϵ	ϵ	ϵ	.01	.99

Gradient Mine-Like Transition Probabilities						
State	1	2	3	4	5	6
1	.59	.17	.01	.07	.11	.05
2	.18	.65	.01	.09	.05	.02
3	.01	.01	.96	.01	.01	.01
4	.37	.46	.01	.08	.05	.03
5	.57	.23	.01	.07	.07	.05
6	.54	.24	.01	.08	.09	.04

Gradient Non-Mine-Like Transition Probabilities						
State	1	2	3	4	5	6
1	.11	.14	.01	.16	.54	.04
2	.11	.13	.01	.23	.48	.05
3	.16	.16	.16	.16	.18	.17
4	.06	.08	.01	.76	.08	.01
5	.14	.13	.01	.09	.58	.06
6	.02	.03	.01	.01	.08	.86

Viterbi and batch gradient training for each HMM, one for mine-like and one for the non-mine-like classes. In this table, ϵ indicates a quantity on the order of 10^{-12} or smaller. An ϵ is inserted during the Viterbi alignment so that if no transitions occur from one state to another in the training set, the sequence probability will not be zero rather it takes a very small quantity. This will not set the probability of the observation sequence to zero for one bad transition. Looking at the mine-like transitions in the Viterbi method, it can be seen that if the output of the MLP at the previous time was at state 1 (object 1), there is a 98% chance of returning to the same state, and a 1% chance of moving to either state 2 or state 5. This is nearly ideal, as there is only a 1% chance that the model would encounter a misclassification (moving to a non-mine-like object, i.e., states 4–6). On the other hand, if the state is 4 (a misclassification), the state could move to any other state with equal probability. It would be desirable that with a misclassification, the transition probabilities guided the model back to a correct classification, and that the model allows for misclassifications that are inevitable. Looking at the corresponding entries for the mine-like transitions in the batch gradient descent method, it can be seen that for state 1, there is a 23% chance of a misclassification (states 4–6) and if a misclassification occurs in state 4, there is an 84% chance the model returns to a correct classification (states 1–3). Indeed the gradient descent is much more flexible when encountering a new environmental condition, as it is more tolerant of transitions that did not occur often in the training set. The validation and testing sets will not have the same transitions as the training set and, hence, we would like to have a model which penalizes the transitions that do not occur in the training set, but does not over-penalize. With many transitions in the Viterbi-trained transition probabilities, a single transition not seen in the training set will cause an extremely low-sequence probability. This is undesirable since transitions between two objects in the same class are bound to occur and the Viterbi-trained matrices do not account for this as well as the proposed batch gradient descent method. If the data was ideal, the Viterbi algorithm would give transition probabilities that are ideal for the data. With an extremely small transition probability ϵ, the emission probabilities found by the MLP have a much smaller impact on the overall multi-aspect sequence classification. The MLP is a discriminatory classifier and as such we would like to keep a proper emphasis on its outputs. The gradient descent method trains to a local minimum which allows for transitions not found in the training set and transitions between objects in the same class. This offers better performance on data sets that vary slightly from the training data.

D. Prediction-Based HMM

The prediction-based HMM in Section IV-C was then implemented using the same training sequences used to train the other HMM/MLP systems. A bank of predictors was trained for various choices of AR model order $p \in [1, 6]$. For sequences with observation length more than six, only the most recent six vectors are used in the prediction process. The MLP is the same one used in all the HMM/MLP-based systems. The same mean error vector and covariance matrix are used in the error distribution for all objects in a given class. This is due to the fact that we used two HMM models, one for each class and not for each object.

It is possible to train predictors for each object, although this could degrade the performance, since multiple objects are contained within a single class, and a predictor trained upon a single object may not generalize well when encountering transitions between objects of the same class. This is especially important when moving to a new environmental condition, as a more general predictor would be more effective than several predictors trained specifically on individual objects. Indeed this was the case in this study, as the overall classification rate dropped when more predictors were added, especially in the case of the rough bottom condition. This drop in the performance is attributed to the fact that secondary reflections and other artifacts may cause some of the objects to look like objects of the other class in the training data set.

The overall correct classification rate is plotted against observation sequence length in Fig. 4(f) for the training, validation and testing sets. It is important to note the trend of the

TABLE II
OVERALL CORRECT CLASSIFICATION ON DIFFERENT CLASSIFICATION SCHEMES

Classifier Type	Training	Validation		Testing	
	Mean	Mean	σ	Mean	σ
Discrete HMM	92%	77.8%	.013	72.8%	.012
HMM/MLP trained with Viterbi	99%	87.0%	.012	81.2%	.011
HMM/MLP trained with Gradient Descent	97%	83.0%	.014	78.6%	.010
HMM/MLP trained with Batch Gradient Descent	99%	86.4%	.012	81.6%	.009
Prediction based HMM	99%	82.4%	.010	78.1%	.008
BPNN decision-level Fusion	99%	89.0%	.013	82.3%	.013
Sequential Decision Feedback	99%	87.0%	.010	80.5%	.012

performance on the validation and testing sets. The increase in performance for each of the first few observations is much smaller when compared with the HMM/MLP with the Viterbi training in Fig. 4(b) or the batch gradient descent in Fig. 4(e). The increase in the overall correct classification is about 5% for observation sequences of length 10. The results of the prediction-based method improve slightly as p (order of the predictor) increases. Unfortunately, with the HMM structure any poor prediction (like those for the first observation) is already captured within the model and cannot be corrected for. The prediction becomes least reliable as the normalized mapped error is close to $[1\ 0]$ or $[0\ 1]$. Thus, the performance of the entire system is affected by the error of the first prediction, and at times the probability of a sequence is 0 for the correct class model after the first prediction. There is a limit on how well the prediction-based HMM classifier can perform that is determined largely by how well the vector linear prediction algorithm can predict the state probability vectors. In this case, the predictor is based on a linear model of fixed order, which may not represent the data well enough.

Clearly, the prediction-based HMM scheme did not perform as well on this data set as the HMM/MLP system trained with the Viterbi algorithm in Fig. 4(b). The prediction-based HMM had an overall correct classification rate of 87% and 80% on the validation and testing sets after 10 aspects, respectively. These results represent a decline of 8% and 11%, respectively, when compared with those of the Viterbi algorithm. Additionally, these results after ten aspects are also worse than any of the gradient descent-based methods in Fig. 4(d) and (e) and the sequential decision feedback in Fig. 4(c) while they are comparable to those of the discrete HMM in Fig. 4(a). This is most likely caused by the limitations in the implementation of the prediction algorithm, namely that separate predictors must be trained for each length of the observation sequence and that prediction based upon a single previous aspect $(p = 1)$ is not reliable. It would be possible to use the original transition probabilities until a sufficient number of observations are collected for prediction. In this case, the standard transition probabilities that give the probability of going from one state to another replaces the prediction error that is based upon only one previous vector. Obviously, if the previous vector gives misleading information, the prediction error based on only one previous vector is not as reliable as the transition probabilities. The prediction-based HMM algorithm also loses the sequential nature of the HMM decision making as it only looks at the error in prediction at each step. In this scheme, only six paths within the HMM model are generated, one for each object. With the standard HMM, the Baum–Welch or Viterbi algorithm searches over all possible state paths, or at least the most likely. The probability of a sequence found by each model in the prediction-based HMM uses a constant state sequence, with variable transition probabilities.

Table II shows the overall correct classification rates for three (3) aspects for all the described methods, including the standard decision-level fusion network [23]. This decision-level fusion system uses the same training data and fuses the outputs of a single-aspect three-layer MLP classifier with structure 28-44-28-2 to discriminate between mine-like and non-mine-like objects. The overall correct classification rates on this single-aspect network are 99%, 82%, and 75% on the same training, validation, and testing sets, respectively. To determine the statistical significance of the classification rates on the validation and testing data sets, 20 Monte Carlo trials were performed where different initializations of aspect separation and initial aspects were considered. The mean and standard deviations of the rates are then computed for each data set and each algorithm and the results are presented in Table II.

From this table, some interesting observations can be made. Clearly, the decision-level fusion provides better results on the validation set, but it only combines the individual decisions generated by a single-aspect classifier. As a consequence, the disadvantage of the decision-level fusion is that it only works with a fixed number of aspects, (three in this case) and adding more aspects requires retraining and perhaps even changing the structure of the system. This implies that incorporating new aspects or observations would require training new network structures, which could be very time consuming and impractical. The HMM/MLP with Viterbi and gradient descent-based methods can easily incorporate new aspects as they become available until a high-confidence decision is made. The results in Table II also indicate that both the Viterbi and batch gradient descent-based approaches slightly outperformed the sequential decision feedback [1] as the observation length increased. Finally, it is interesting to note that the prediction-based HMM algorithm offers a high-misclassification rate on targets, but a low false alarm rate. The sequential decision feedback method, the HMM/MLP Viterbi training, and the batch gradient descent methods, on the other hand, provide lower misclassification rates on targets, at a price of higher false alarm rates.

VI. CONCLUSION

In this paper, new approaches for multi-aspect feature-level pattern classification are presented. A new gradient de-

scent-based method is used for training the parameters within the HMM structure. Both online and batch versions of this algorithm are presented. Several connectionist ideas are presented along with how they would be applied to the underwater target classification problem. A hybrid HMM/MLP system that uses the discriminant nature of the MLP outputs to enhance the sequence classification is introduced. A new approach is also described that works within the HMM structure to lift the first-order Markov assumption and improve the computation of the transition probabilities. The method uses a prediction-based HMM using a vector AR process. The prediction error is then mapped and normalized across all classes, giving an estimate of the transition probabilities. This approach allows transition probabilities to reflect the previous states and not just the average transitions found across the entire training set. The results of different HMM/MLP approaches are presented and benchmarked on a wideband sonar data set for underwater target classification. The discrete HMM is found to be inferior to all the other methods tried owing to the inherent quantization that severely degrades the performance. The three HMM/MLP-based approaches provided much better performance than the discrete HMM. The results showed that the HMM/MLP trained with forced Viterbi training is effective, but is somewhat limited when moving from one environment to another. The batch gradient descent-based training provided comparable results to those of the Viterbi training while offering better flexibility in dealing with new environments. The prediction-based HMM is a very promising method though is limited at the beginning of a sequence by insufficient data to perform the prediction. Overall, hybrid HMM/MLP approach is found to be an excellent way to do multi-aspect pattern classification. The HMM does a good job of overall classification, while the MLPs estimates of the emission probabilities improve that performance with better single-aspect decisions. Thus, a combination of MLP with HMM training is a natural and promising way of performing multi-aspect classification using feature-level fusion.

ACKNOWLEDGMENT

The authors would like to thank the NSWC-Coastal Systems Station, Panama City, FL, and the Applied Research Labs, University of Texas (ARL-UT), Austin, under the auspices of the Office of Naval Research, Code 321 Undersea Signal Processing, for providing the data and technical support.

REFERENCES

[1] M. Azimi-Sadjadi, A. A. Jamshidi, and G. J. Dobeck, "A new sequential decision feedback method with application to target classification," in *Proc. 2002 IEEE Int. Joint Conf. Neural Networks*, vol. 2, pp. 697–702.

[2] H. Borotschnig, L. Paletta, M. Prantl, and A. Pinz, "A comparison of probabilistic, possibilistic and evidence theoretic fusion schemes for active object recognition," *Computing*, vol. 62, pp. 293–319, 1999.

[3] J. Denzler and C. Brown, "Information theoretic sensor data selection for active object recognition and state estimation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 2, pp. 145–157, Feb. 2002.

[4] B. V. Dasarathy, "Sensor fusion potential exploitation-innovative architectures and illustrative applications," *Proc. IEEE*, vol. 85, no. 1, pp. 24–38, Jan. 1997.

[5] P. K. Varshney and C. S. Burrus, Eds., *Distributed Detection and Data Fusion (Signal Processing and Data Fusion)*, 1st ed. New York: Springer-Verlag, 1997.

[6] M. Robinson, J. Salazar, and M. R. Azimi-Sadjadi, "Multi-aspect pattern classification using predictive networks and error mapping," in *Proc. 2002 IEEE Int. Joint Conf. Neural Networks*, vol. 2, May 2002, pp. 1842–1847.

[7] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proc. IEEE*, vol. 77, no. 2, pp. 257–286, Feb. 1989.

[8] S. W. Perry and L. Guan, "Detection of small man-made objects in sector scan imagery using neural networks," in *Proc. IEEE Int. Conf. Oceanic Engineering*, Honolulu, HI, Nov. 2001, pp. 2108–2114.

[9] I. T. Ruiz, D. M. Lane, and M. J. Chantler, "A comparison of inter-frame feature measures for robust object classification in sector scan sonar image sequences," *IEEE J. Ocean. Eng.*, vol. 24, no. 4, pp. 458–469, Oct. 1999.

[10] M. J. Chantler, D. M. Lane, D. Dai, and M. Williams, "Detection and tracking of returns in sonar scan image sequences," in *Proc. IEE Radar Sonar Navigation*, vol. 143, Jun. 1996, pp. 157–162.

[11] H. Bourlard and C. Wellekens, "Links between Markov models and multilayer perceptrons," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 12, no. 12, pp. 1167–1178, Dec. 1990.

[12] N. Morgan and H. Bourlard, "Continuous speech recognition," *IEEE Signal Process. Mag.*, vol. 12, no. 3, pp. 25–42, May 1995.

[13] H. Misra, H. Bourlard, and V. Tyagi, "New entropy based combination rules in HMM/ANN multi-stream ASR," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing (ICASSP '03)*, vol. 2, Apr. 6–10, pp. 741–744.

[14] P. R. Runkle, P. K. Bharadwaj, and L. Carin, "Target identification with wave-based matched pursuits and hidden Markov models," *IEEE Trans. Antennas Propag.*, vol. 47, no. 10, pp. 1543–1554, Oct. 1999.

[15] S. Mallat and Z. Zhang, "Matched pursuits with time-frequency dictionaries," *IEEE Trans. Signal Process.*, vol. 41, no. 12, pp. 3397–3415, Dec. 1993.

[16] G. D. Forney, Jr., "The Viterbi algorithm," *Proc. IEEE*, vol. 61, no. 3, Mar. 1973.

[17] L. R. Rabiner, S. E. Levinson, and M. M. Sondhi, "An introduction to the application of the theory of probabilistic functions of a Markov process to automatic speech recognition," *Bell Syst. Tech. J.*, vol. 62, pp. 1035–1074, Apr. 1983.

[18] S. E. Levinson, B. H. Juang, and M. M. Sondhi, "Maximum likelihood estimation for multivariate mixture observations of Markov chains," *IEEE Trans. Inf. Theory*, vol. IT–32, no. 2, pp. 307–309, Feb. 1986.

[19] S. Renals *et al.*, "Connectionist probability estimators in HMM speech recognition," *IEEE Trans. Speech and Audio Process.*, vol. 2, no. 1, pp. 161–174, Jan. 1994.

[20] L. Xu and M. Azimi-Sadjadi, "Parameter estimation for two-dimensional vector models using neural networks," *IEEE Trans. Signal Process.*, vol. 43, no. 12, pp. 3090–3094, Dec. 1995.

[21] S. Haykin, *Adaptive Filter Theory*, 2nd ed. Englewood Cliffs, NJ: Prentice-Hall, 1991.

[22] "Target data acquistion setup," ARL-UT Facility, Internal documentation, ARL-UT, 1998.

[23] M. R. Azimi-Sadjadi, D. Yao, Q. Huang, and G. J. Dobeck, "Underwater target classification using wavelet packets and neural networks," *IEEE Trans. Neural Netw.*, vol. 11, no. 3, pp. 784–794, May 2000.

[24] Y. Linde, A. Buzo, and R. M. Gray, "An algorithm for vector quantizer design," *IEEE Trans. Commun.*, vol. 28, no. 1, pp. 84–95, Jan. 1980.

Marc Robinson received the B.S. degree in electrical engineering from Brigham Young University, Provo, UT, in 1999, and the M.S. degree in electrical engineering from Colorado State University, Fort Collins, in 2003, with specialization in pattern recognition and neural networks.

He is currently employed at Signal Systems Corporation, Severna Park, MD. His research interests include neural networks, pattern recognition, acoustics, and sonar.

Mahmood R. Azimi-Sadjadi (M'81–SM'89) received the M.S. and Ph.D. degrees in electrical engineering with specialization in digital signal/image processing from the Imperial College of Science and Technology, University of London, U.K., in 1978 and 1982, respectively.

He is currently a Full Professor in the Electrical and Computer Engineering Department, Colorado State University (CSU), Fort Collins. He is also serving as the Director of the Digital Signal/Image Laboratory at CSU. His main areas of interest include digital signal and image processing, target detection, classification and tracking using broadband sonar, radar and IR systems, adaptive filtering and system identification, and neural networks. His research efforts in these areas resulted in over one hundred seventy journal and refereed conference publications. He is a coauthor of the book *Digital Filtering in One and Two Dimensions* (New York: Plenum, 1989).

Dr. Azimi-Sadjadi was the recipient of the 1999 ABELL Teaching Award, the 1993 ASEE-Navy Senior Faculty Fellowship Award, the 1991 CSU Dean's Council Award, and the 1984 DOW Chemical Outstanding Young Faculty Award. He served as an Associate Editor for the IEEE TRANSACTIONS ON SIGNAL PROCESSING and the IEEE TRANSACTIONS ON NEURAL NETWORKS.

Jaime Salazar received the B.S. degree in physics and the M.S. degree in computer science from the Instituto Teconologico y de Estudios Superiores de Monterrey, Monterrey, Mexico, in 1981 and 1991, respectively. He is currently working toward the Ph.D. at the Electrical and Computer Engineering Department, Colorado State University, Fort Collins.

His research areas of interest include pattern recognition, neural networks, text and image retrieval systems, and image processing.